Comparison of different implementations of the first stage of the band reduction

This work aims to investigate different ways to implement the first stage of the band reduction algorithm then select the best before moving to the second stage. Basically we identified three strategies:

- Panel oriented strategy: factorize (QR) the whole tile-panel before updating the trailing matrix, and similarly for LQ. This is strategy used in the old plasma (QUARK).
- Tile oriented strategy: perform update on the trailing matrix in parallel with panel factorization, and similarly for LQ.
- Tile oriented for QR and panel oriented for LQ: apply the tile oriented strategy for QR and panel the oriented strategy for LQ.

Furthermore, currently the Plasma supports dependency only between full tiles, while some kernels (tsqrt and zunmqr, for example) only require read and write dependency on the triangular half of some tiles. We can increase the parallelism by providing a strategy to express the dependencies on lower/upper triangular tiles. To achieve this, we modify the data dependency in an unsafe way.

We implemented all these strategies and compare them with the QUARK version. All these kernels achieve similar performance to QUARK (in terms of time). However the performance in terms of Gflops seems significantly lower than QR factorization. We are still investigating the causes of this issue in all the versions including QUARK. It may have something to do with the way we count the flops.

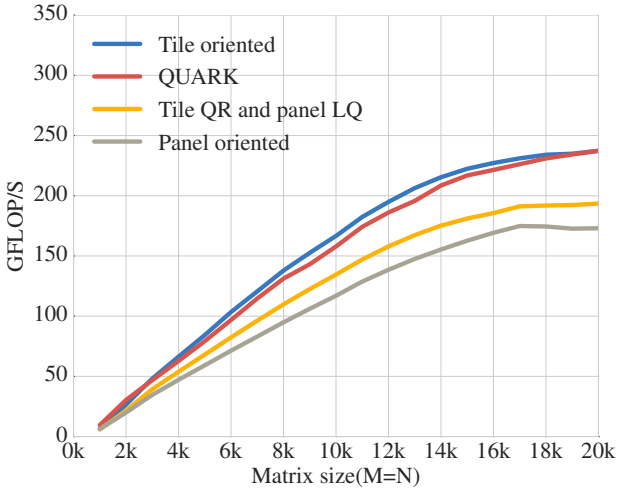
As illustrated in Figure 1a, the tile oriented version is outperforming the other strategies. But by expressing some dependency at lower/upper triangular tile granularity (Figure 1b), both the panel oriented and mixed panel and tile oriented strategies improved to achieve the same performance as the tile oriented version.

The same behaviour is observed when data are allocated in the high bandwidth memory (HBW), instead of the regular DDR4 RAM, except the fact that the performance improved for each version.

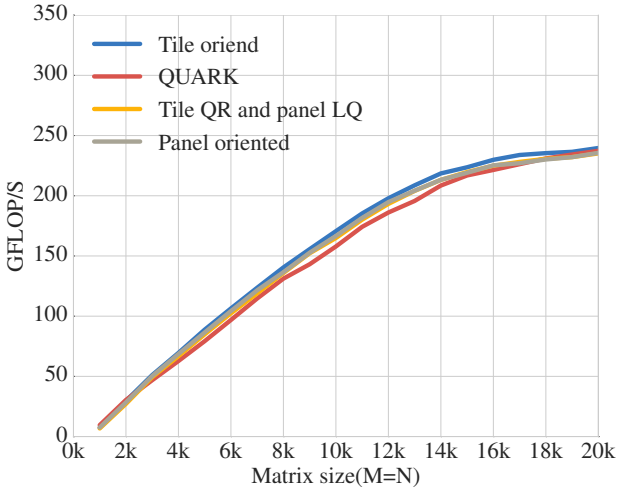
From this preliminary experiment, we observed that the tile oriented strategy seems to be the best option. While the other strategies benefit from the data dependency modification, the tile oriented strategy doesn't seem to show significant improvement.

We have three scenarios for the implementation of the second stage:

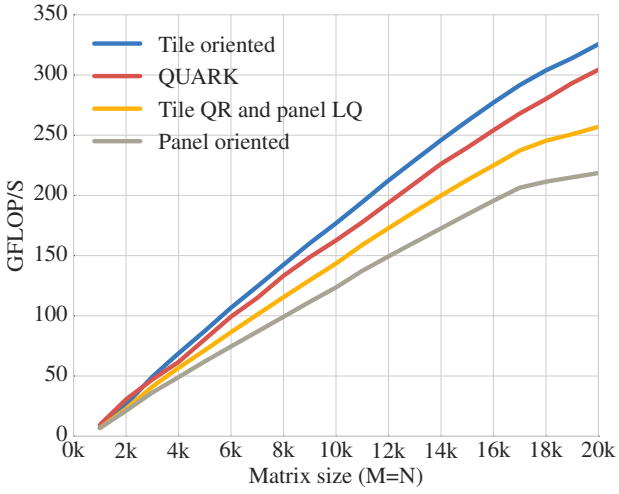
1. Use the MKL GBBRD kernel for reduction from band to bidiagonal (almost finished)
   - Convert the tile band matrix (AB) to LAPACK band format
   - Use the MKL GBBRD kernel: AB = U2* Bidiag* VT2
   - Convert the U2 and VT2 to tile layout then accumulate the transformation from the first stage
   - Convert the accumulated transformations to LAPACK layout then use MKL for the SVD of the bidiagonal matrix
2. Clean up the second stage of the old plasma (seems very messy)
3. Possibly collaborate with Bo Kagstrom et. al. for a new bulge chasing algorithm



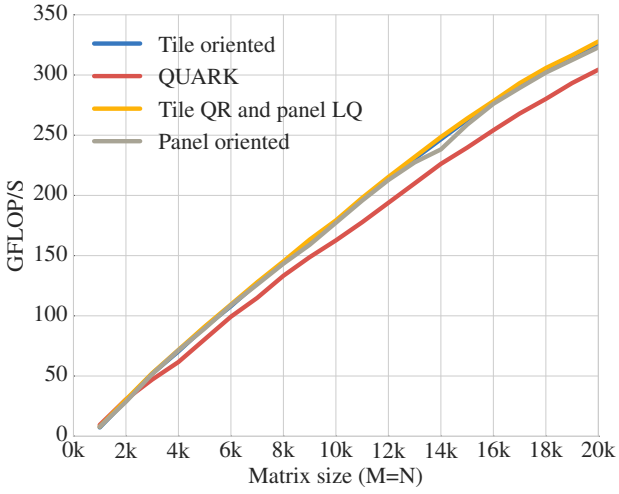
(a) **DDR4 memory** with all data dependencies at `square tile` granularity.



(b) **DDR4 memory** with some data dependencies at `lower/upper triangular tile` granularity.



(c) **HBW memory** with all data dependencies at `square tile` granularity.



(d) **HBW memory** with some data dependencies at `lower/upper triangular tile` granularity.

Figure 1: Performance comparison of different implementations of DGE2GB using 68 threads on the Intel KNL with different square matrices ranging in size from $1,000 \times 1,000$ to $20,000 \times 20,000$.